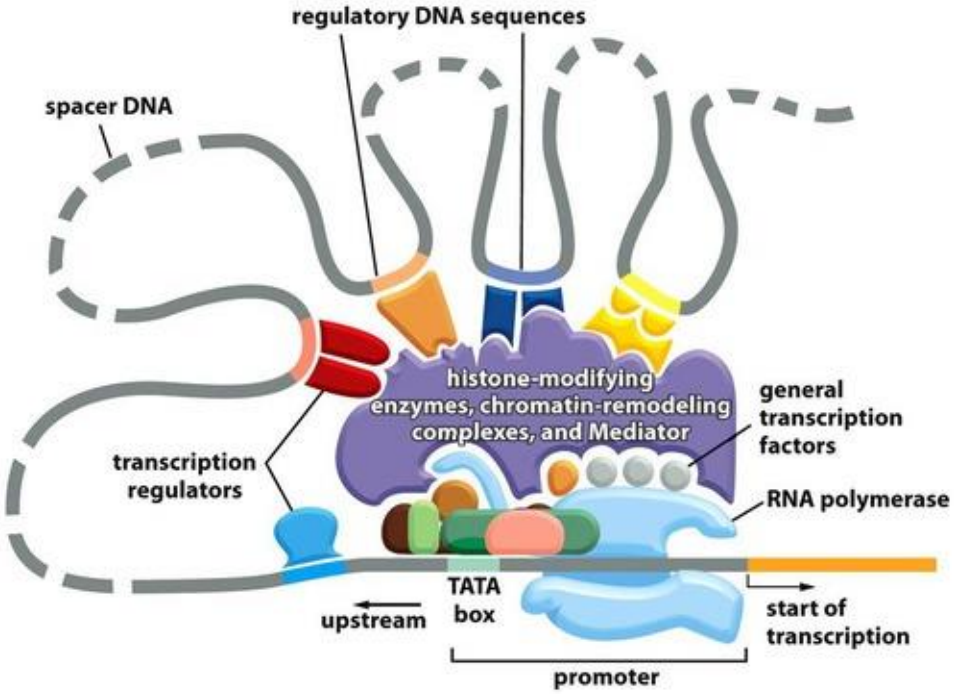
DEEP LEARNING FOR GENOMICS

Raphaël MOURAD, Assist. Prof.
Visiting prof. INRAe
Université Paul Sabatier, Toulouse III

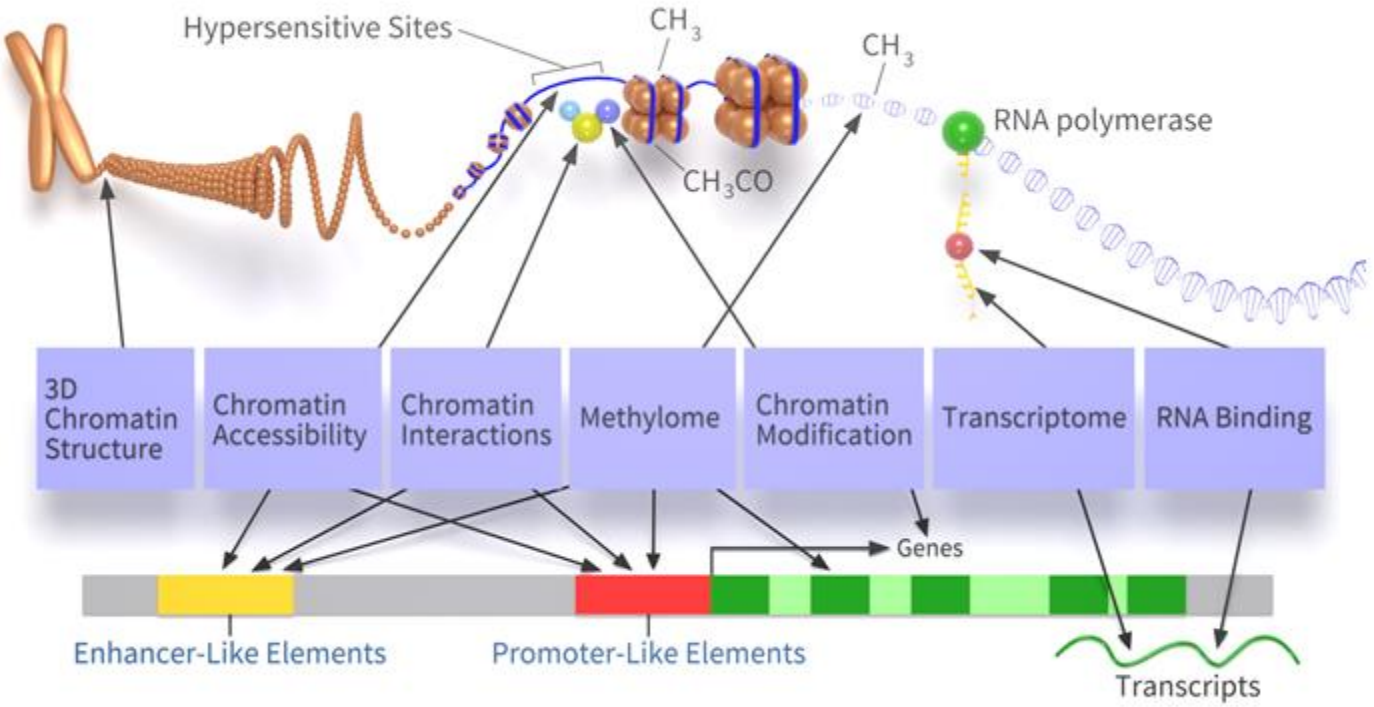
CLASSIFICATION OF DNA SEQUENCES USING MACHINE/DEEP LEARNING

Regulatory regions



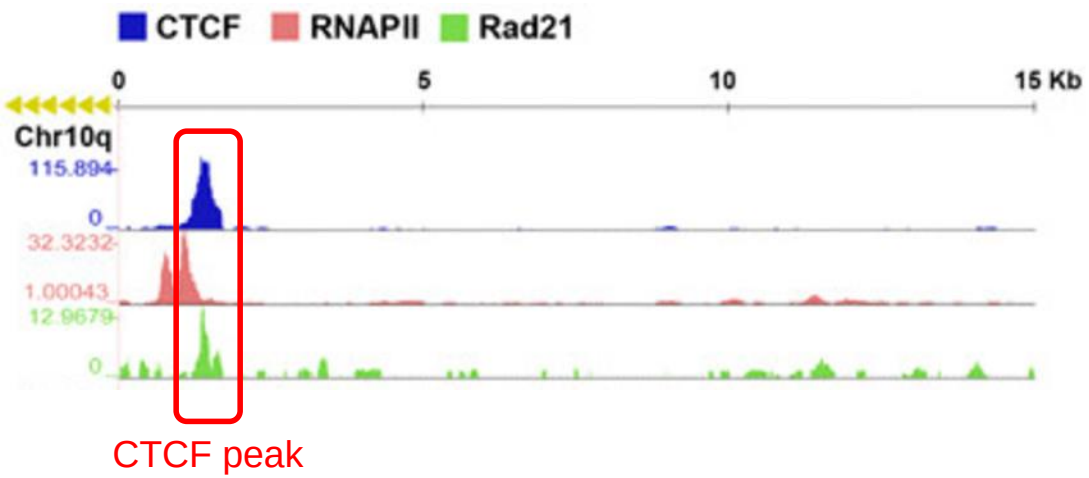
- Regulatory regions (promoters, enhancers, insulators, …) are non-coding DNA sequences that control the expression of target genes.

Regulatory regions

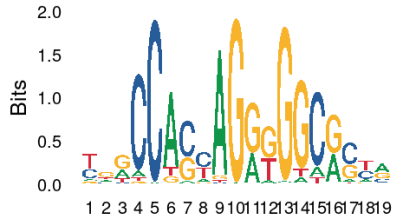


- Regulatory regions were mapped during the last decade using techniques such as ChIP-seq, ATAC-seq, Hi-C, methyl-seq...

CTCF ChIP-seq peaks as examples



- We extract the sequences of the CTCF ChIP-seq peaks.
- If we run a motif search (using MEME for instance), we will observe the CTCF motif MA0139.1:



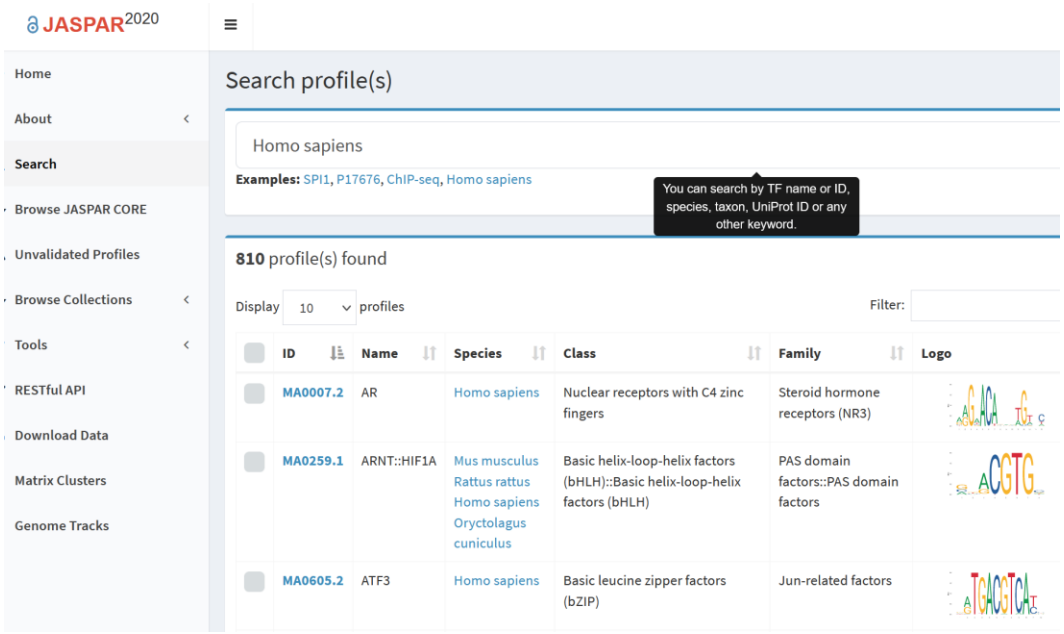
Classifier to distinguish CTCF bound sequences from other sequences

- One can build a simple classifier that distinguish between CTCF bound sequences and other sequences by looking for the occurrence of the motif MA0139.1.

- NB: other sequences (negative sequences) can be:
  - Shuffled bound sequences while conserving dinucleotides (ie 2nd order markov model) as implemented in MEME suite.
  - Real unbound sequences drawn from the genome with GC content, length and mappability similar to the bound sequences as implemented in gkmSVM R package.

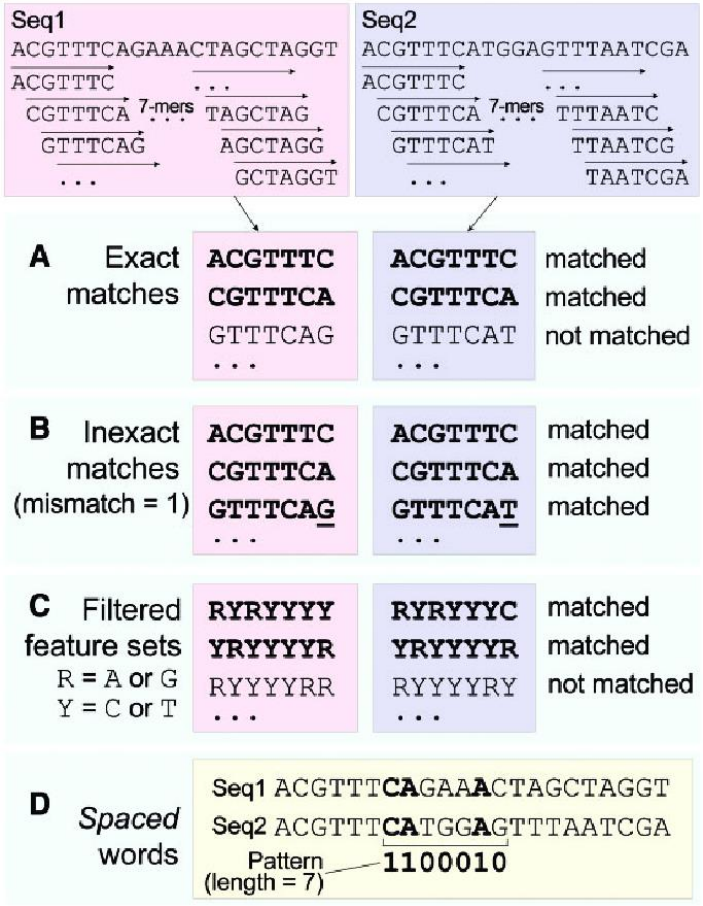
How to obtain features from the DNA sequences?

- Known transcription factor motifs (JASPAR, HOCOMOCO databases): motif occurrence can be used.
- Very easy but you will miss features that are important predictors but are not in the database!



How to obtain features from the DNA sequences?

- K-mers (for instance: ATCTC or ATTTC, …): very powerfull approach since almost no prior information (except k) is needed to build the features. Then you can count the occurrence of every k-mer.



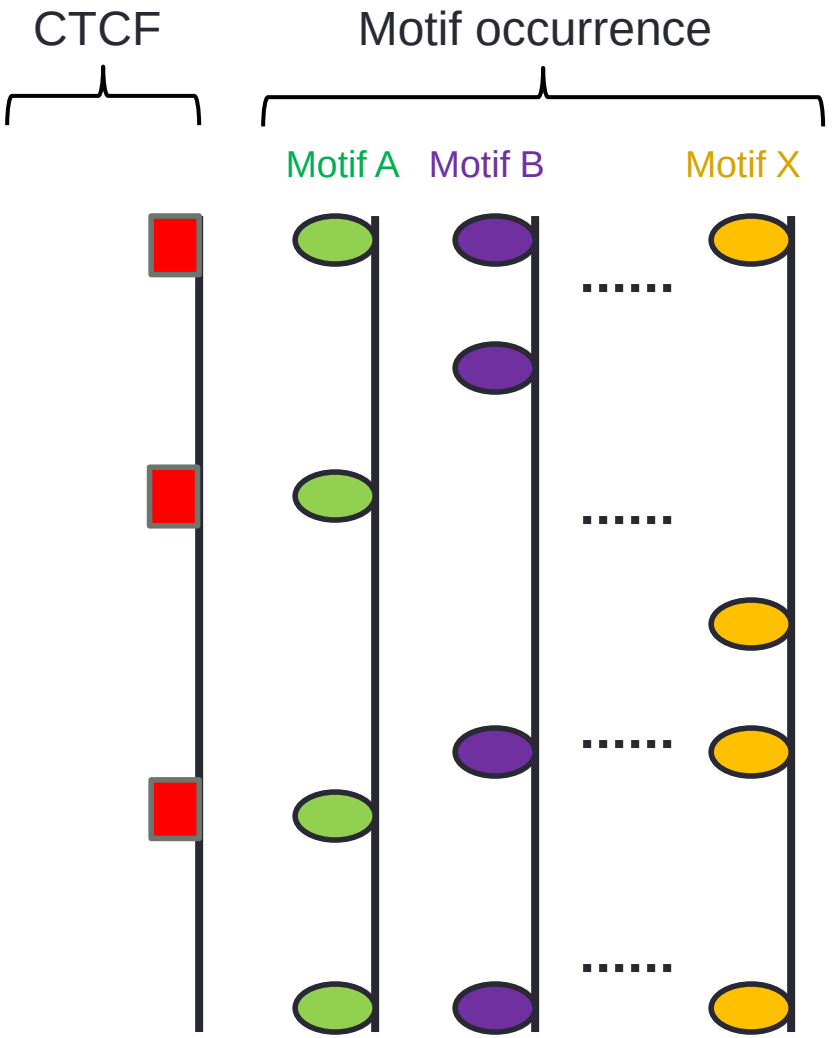
How to obtain features from the DNA sequences?

- Many other possible features:
  - Di/trinucleotide composition
  - DNA shape
  - Microsatellites: any kind of repeats
  - DNA conservation among species (PhastCons score)
  - Any other motif from the litterature.

How to obtain features from the DNA sequences?

- Many other possible features:
  - Di/trinucleotide composition
  - DNA shape
  - Microsatellites: any kind of repeats
  - DNA conservation among species (PhastCons score)
  - Any other motif from the litterature.

Data table



Model variables

Y	X1	X2	…	Xp
1	2	1	…	5
0	0	0	…	0
0	0	1	…	0
0	0	0	…	0
1	1	0	…	0
0	0	0	…	0
0	0	0	…	2
0	0	0	…	0
0	0	3	…	7
1	1	0	…	0
0	0	0	…	0
0	0	0	…	0
1	1	2	…	1

Simple model: logistic regression

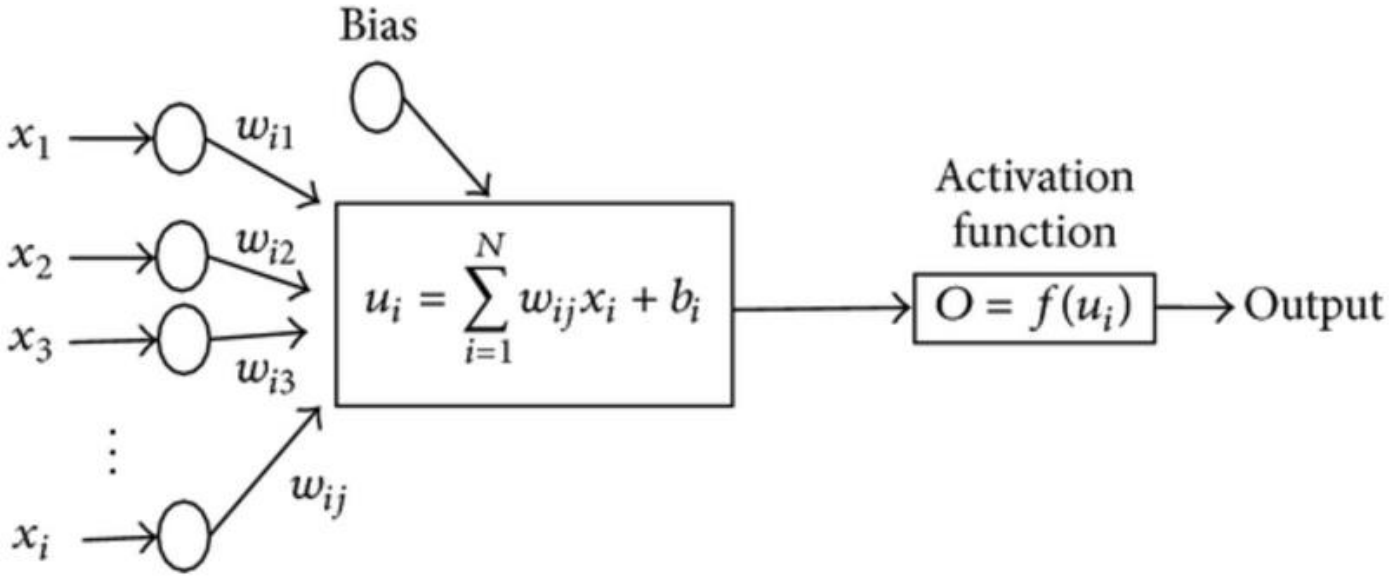
Logistic regression

$$ln\frac{Prob(Y=1|X)}{1-Prob(Y=1|X)}=\beta_0+\beta X$$

- One of the most basic classifier.
- Assumes additivity effect of predictors (each predictor contributes independently).
- Can add lasso penalty to improve predictions.

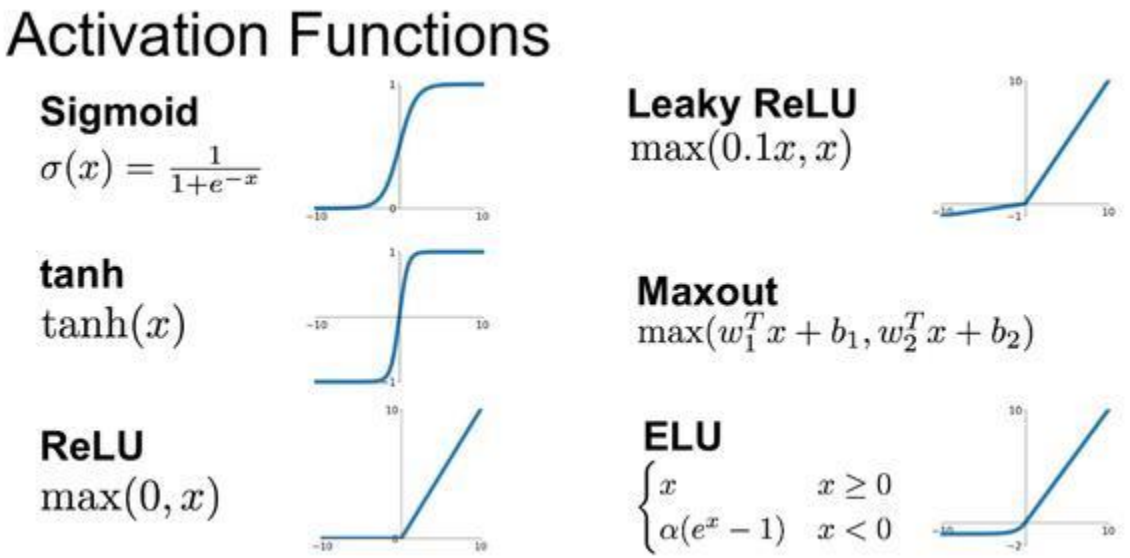
SHORT INTRO TO DEEP LEARNING

Neural networks



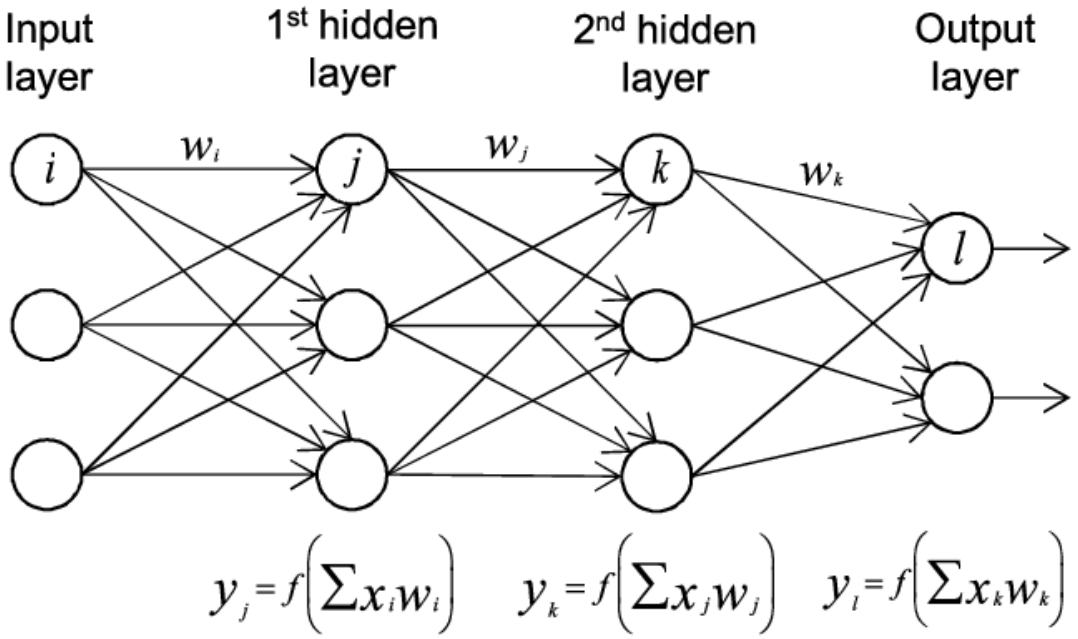
- In a neural network, multiple inputs xi are combined through a linear combination (with weights wi), and then an activation function is used for a non-linear transformation to obtain the output.

Activation function



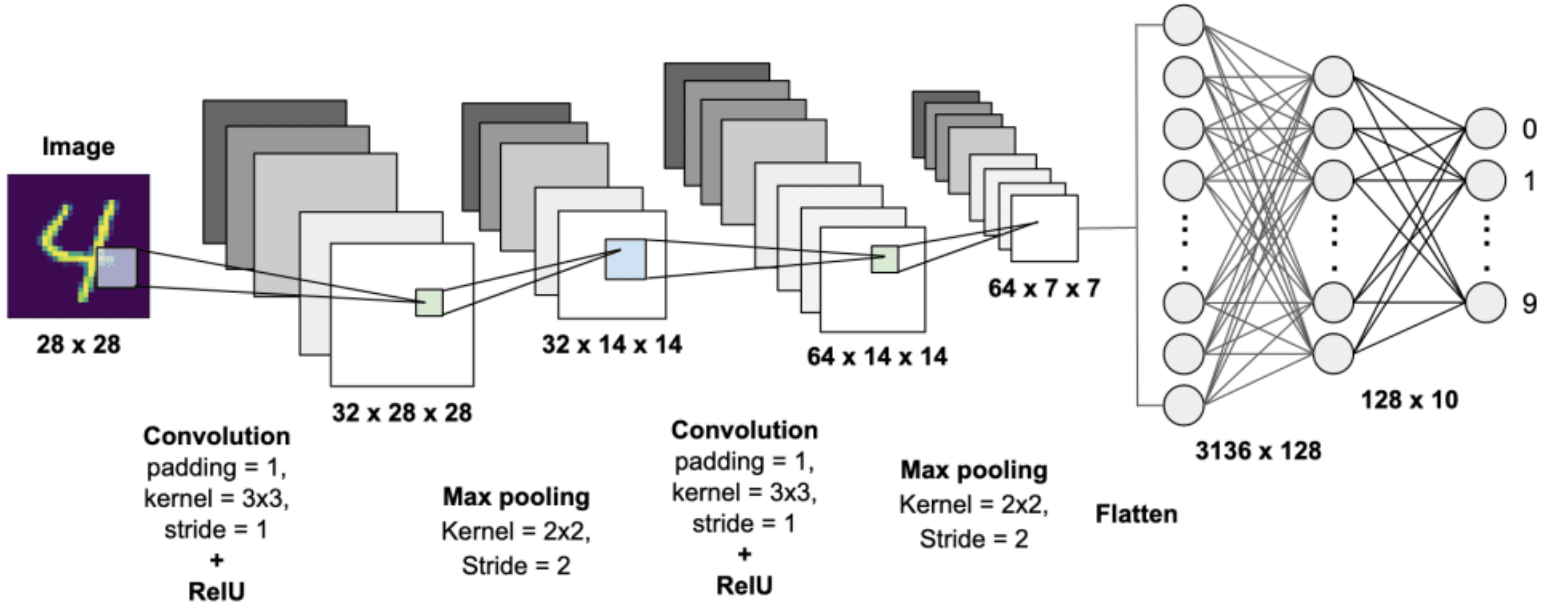
- The activation function allows to obtain a non-linear output from a linear input.
- NB: the linear activation function also exists (A = cx).

Deep neural networks



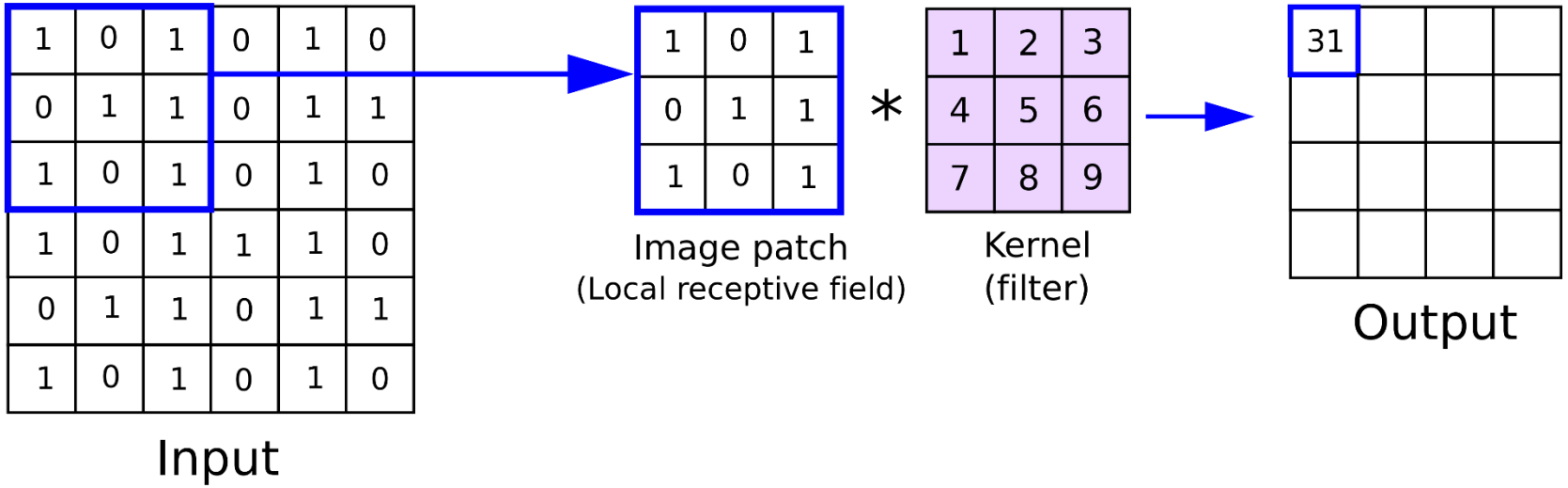
- A deep neural network (DNN) is an neural network (NN) with multiple layers between the input and output layers. Each hidden layer linearly combines the output from the previous layer and then does a non-linear transformation.

Convolutional neural networks (CNNs)



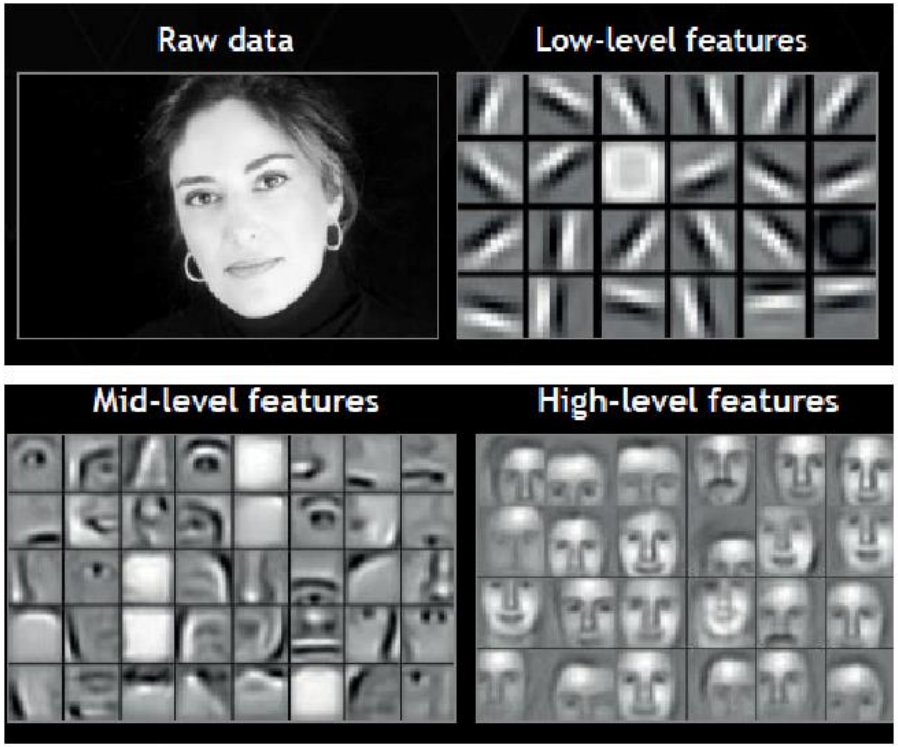
- A CNN is based on the stacking of one or more convolutional layers. A convolutional layer is based on the shared-weight architecture of the convolution kernels (or filters) that slide along input features and provide translation equivariant responses known as feature maps.

Convolutional layer



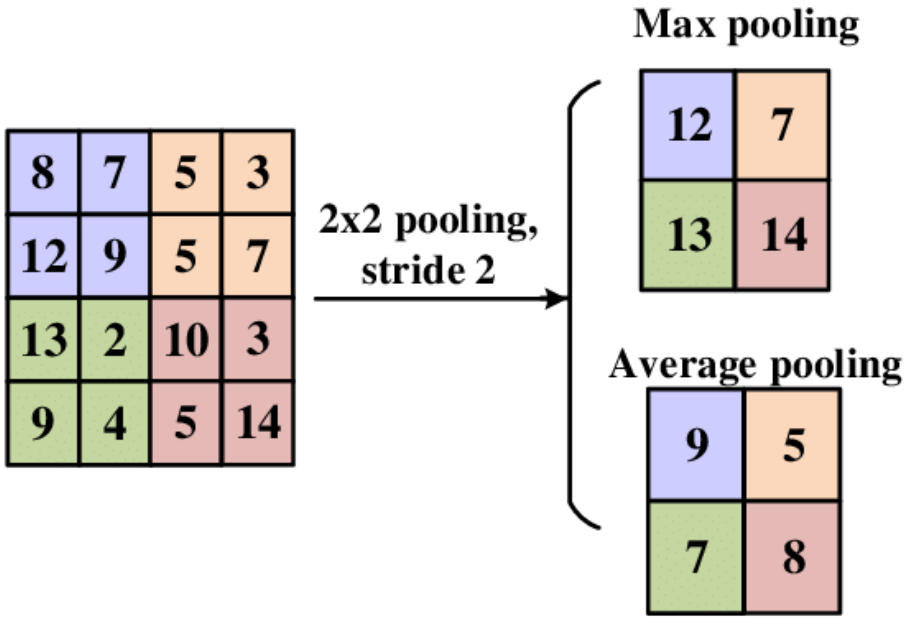
- A patch (submatrix) in the input matrix is multiplied by a kernel (or filter) to obtain an output value. This operation is done for every patch to obtain every output value.

What do the kernels represent in the convolutional layer(s)?



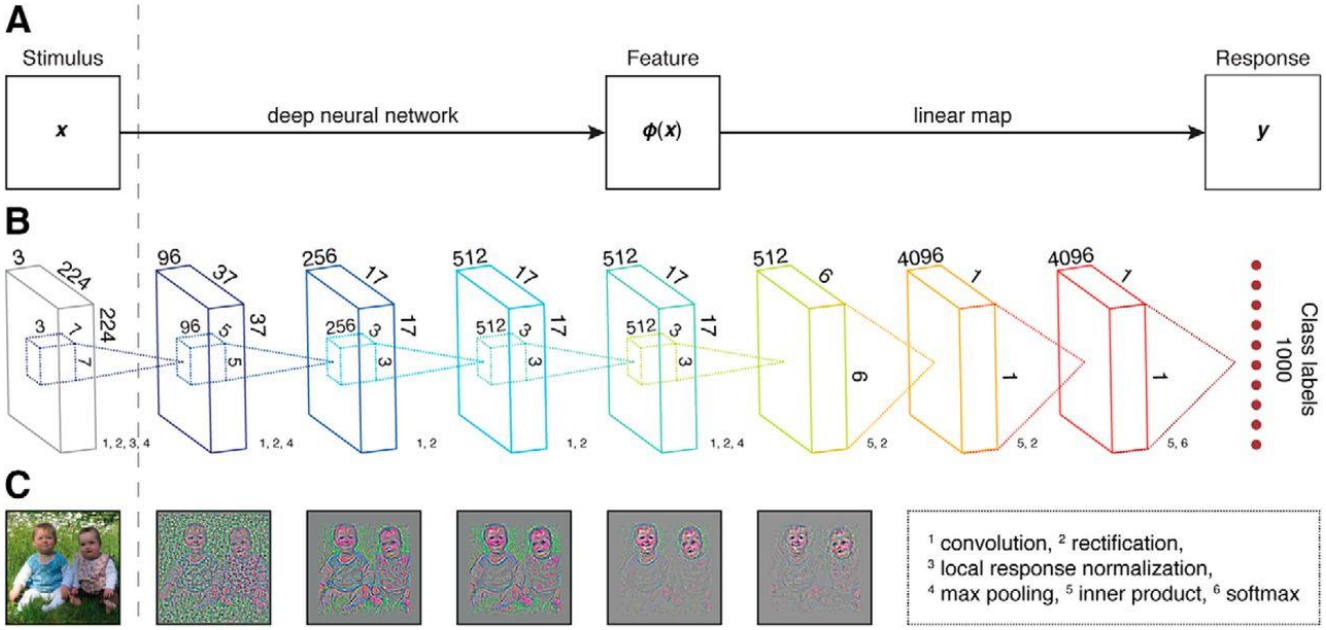
- In the first conv layer, the kernels correspond to low-level features (often edges). In the middle conv layers, the kernels correspond to mid-level features (parts of an object). In the last conv layers, the kernels correspond to high-level features (often objects).

Pooling layer



- The pooling layer aims to reduce the dimension of the input matrix in order to summarize the underlying information.

Principle of deep learning : stacking many different sorts of layers



- Building a deep neural network is like assembling a lego toy where every lego brick is a layer.

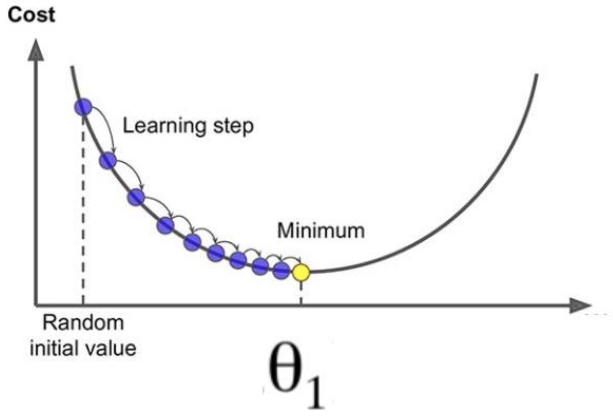
Parameter learning by gradient descent

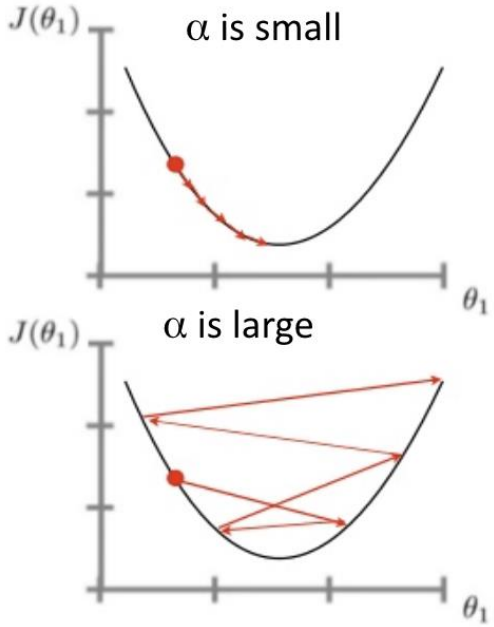
repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

}





Loss functions

- For regression (continuous outputs):
  - Mean Square Error:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

  - Mean Absolute Error:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\boldsymbol{x}_i - \boldsymbol{x}|$$

Loss functions

- For classification (categorical outputs):
  - Cross entropy:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

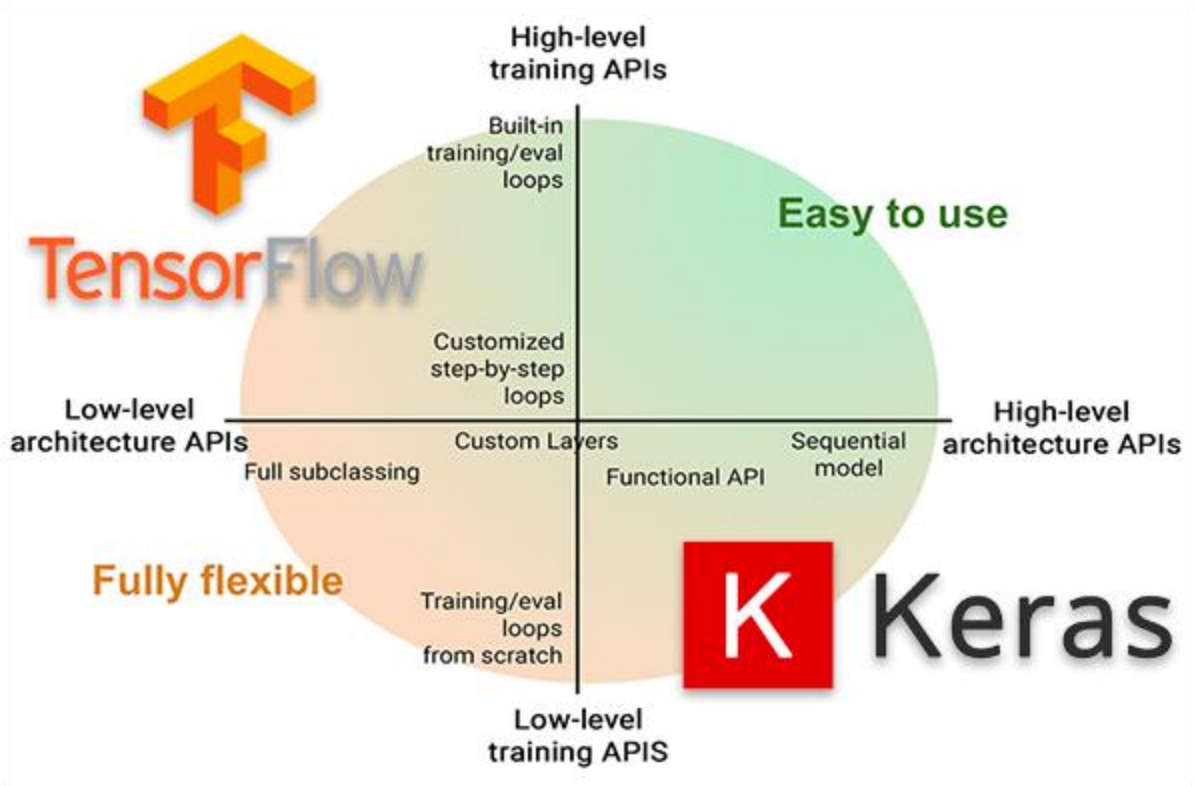
Binary Cross-Entropy / Log Loss

TENSORFLOW AND KERAS

Tensorflow

- Tensorflow is a library developed by Google to develop and train deep neural networks. Version 1.0.0 was released on February 11, 2017.
- Run on multiple CPUs and GPUs (or now on TPUs).
- Python library, but can be used in other languages such as R.
- Very fast and memory efficient library.
- Very flexible library: you can implement any function and make your own layers for deep learning.

Keras as a simplification of Tensorflow



Keras

- Easy to implement a model in a just a few lines of code.

Build the model

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1600)              0
_________________________________________________________________
dropout (Dropout)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
```

Simple MNIST convnet

- Setup
- Prepare the data
- Build the model
- Train the model
- Evaluate the trained model

Keras: two modes

- tf.keras.Sequential() class:
  - Every line corresponds to a layer.
  - Easy to code.
  - Cannot deal with complex models with multiple inputs and outputs!

- tf.keras.Model() class:
  - More complex models with multiple inputs and outputs.
  - Can be used to implement any model.
  - Not for beginners!

TF and keras installation

- For CPU installation (computation is slow), installation is easy!

- For GPU installation (computation is optimized and uses GPU), installation is quite tricky!
  - Need to install python, CUDA, Cudnn, tensorflow, Keras, with versions that are compatible with your graphic card.

Keras R

R interface to Keras

R-CMD-check passing | CRAN 2.4.0 | license MIT

Keras is a high-level neural networks API developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.* Keras has the following key features:

- Allows the same code to run on CPU or on GPU, seamlessly.
- User-friendly API which makes it easy to quickly prototype deep learning models.
- Built-in support for convolutional networks (for computer vision), recurrent networks (for sequence processing), and any combination of both.
- Supports arbitrary network architectures: multi-input or multi-output models, layer sharing, model sharing, etc. This means that Keras is appropriate for building essentially any deep learning model, from a memory network to a neural Turing machine.

See the package website at https://tensorflow.rstudio.com for complete documentation.

Keras R

```
library(keras)
```

Let's start by loading and preparing the MNIST dataset. The values of thee pixels are integers between 0 and 255 and we will convert them to floats between 0 and 1.

```
mnist <- dataset_mnist()
mnist$train$x <- mnist$train$x/255
mnist$test$x <- mnist$test$x/255
```

Now, let's define the a Keras model using the sequential API.

```
model <- keras_model_sequential() %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(0.2) %>%
  layer_dense(10, activation = "softmax")
```

Note that when using the Sequential API the first layer must specify the `input_shape` argument which represents the dimensions of the input. In our case, images 28x28.

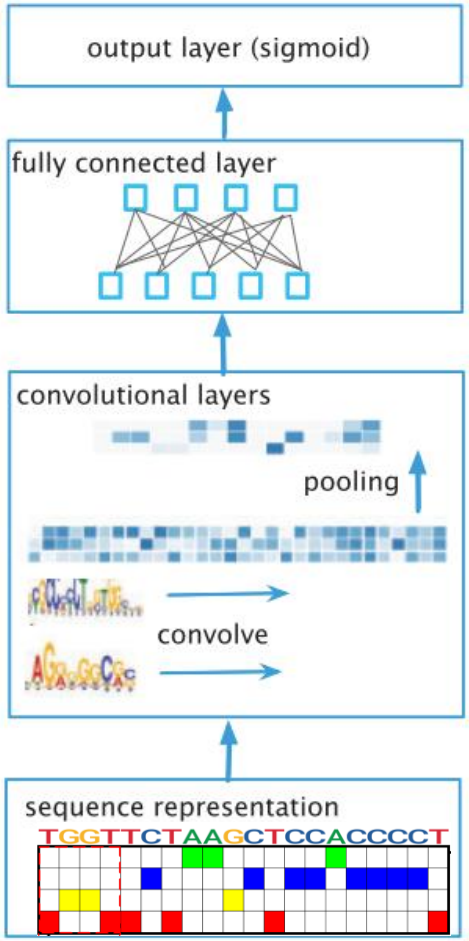
After definning the model, you can see information about layers, number of parameters, etc with the `summary` function:

```
summary(model)
```

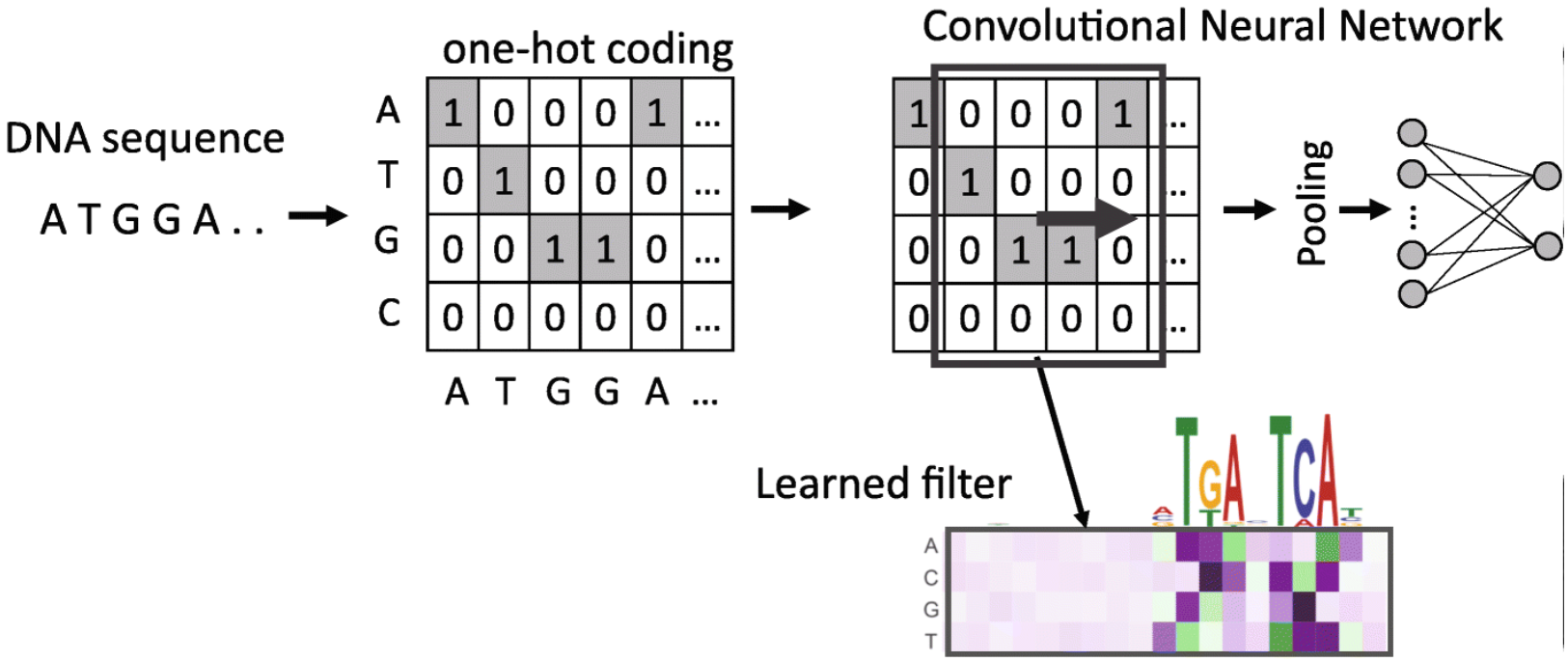
DEEP LEARNING FOR GENOMICS

CNN for classifying DNA sequences

- Binary output
- Standard fully connected layer
- 1-dimension convolution
  = DNA motif scanning
- One-hot encoding of DNA:
  Colored cells = 1; white cells = 0.



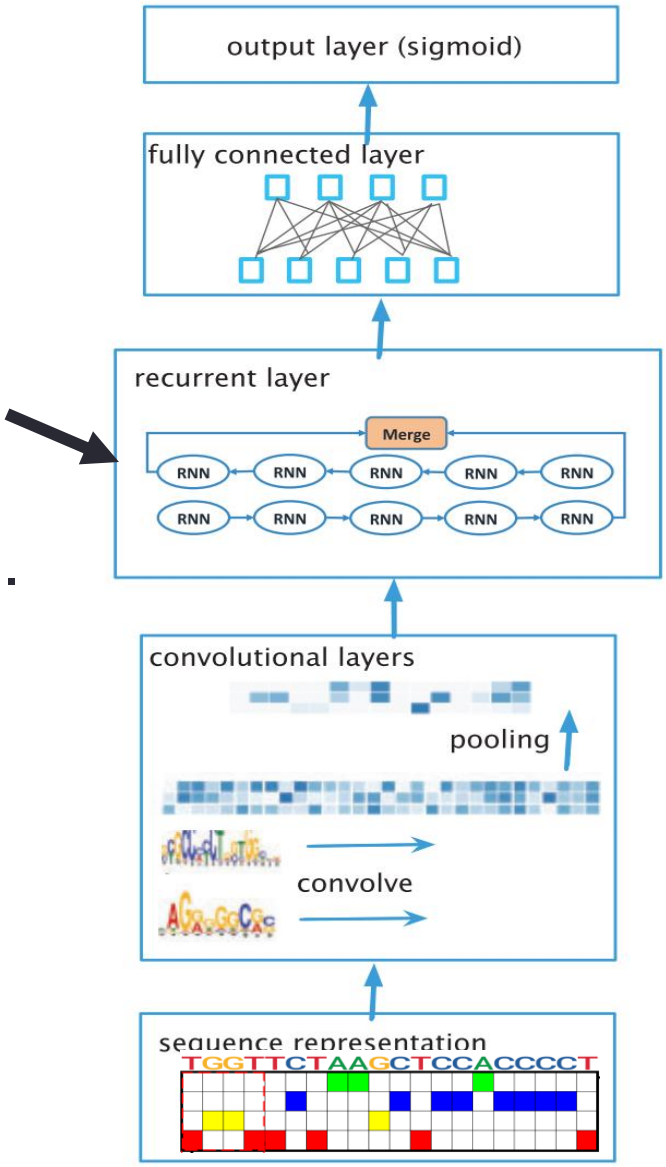
Meaning of 1D-convolution for DNA sequences



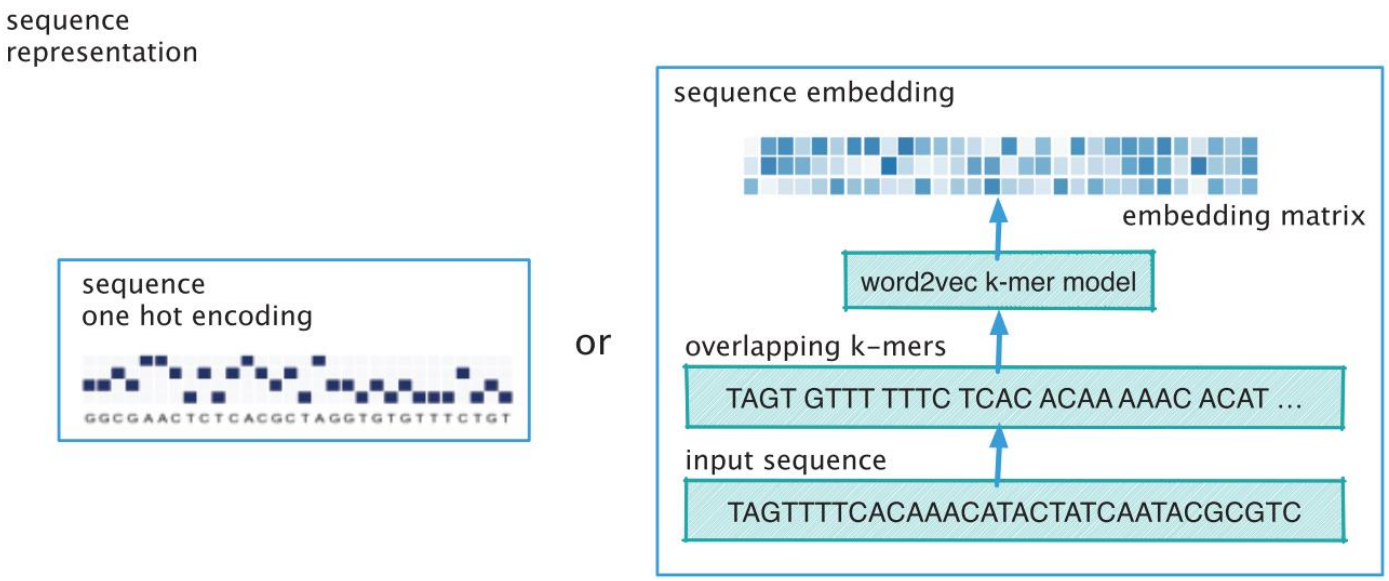
- Based on kernels (filters) that are matrices of weights for each base of a DNA motif.
- Kernels = Position Weight Matrices (given some transformation).

CNN + RNN model

- One can simply add an RNN (or GRU or LSTM or bi-LSTM) layer on the top of the CNN layer in order to capture long-range dependencies between the motifs of the sequence.



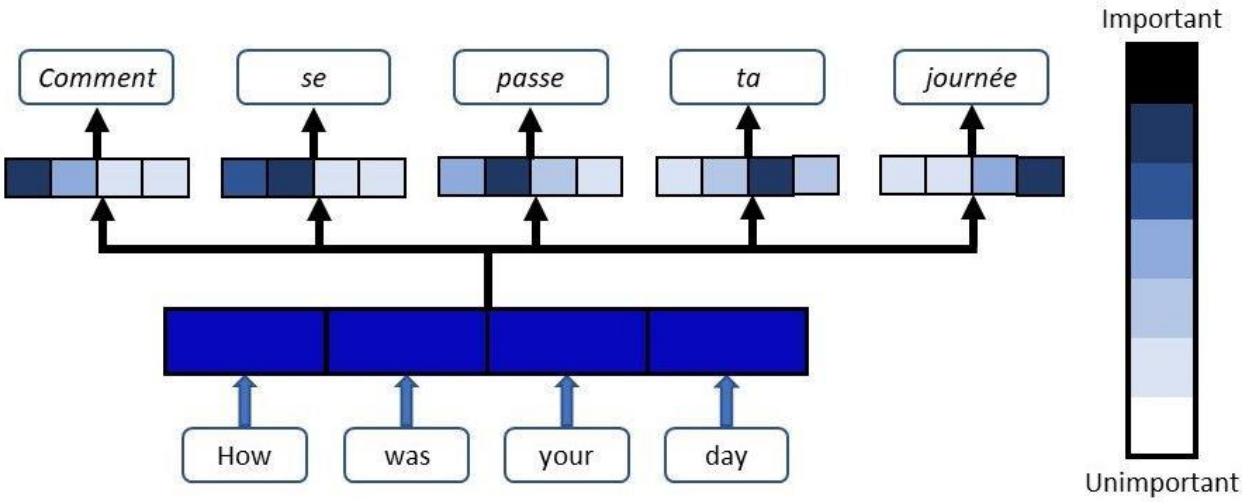
Two options for sequence representations



- 2 options:
  - One hot encoding followed by 1D-convolution,
  - Overlapping k-mers followed by an embedding (as word2vec).

Attention model

- Attention is a technique that mimics cognitive attention.
- Attention has revolutionned natural language processing (and computer vision), and it has replaced convolutional models and RNN (LSTM, GRU) models.



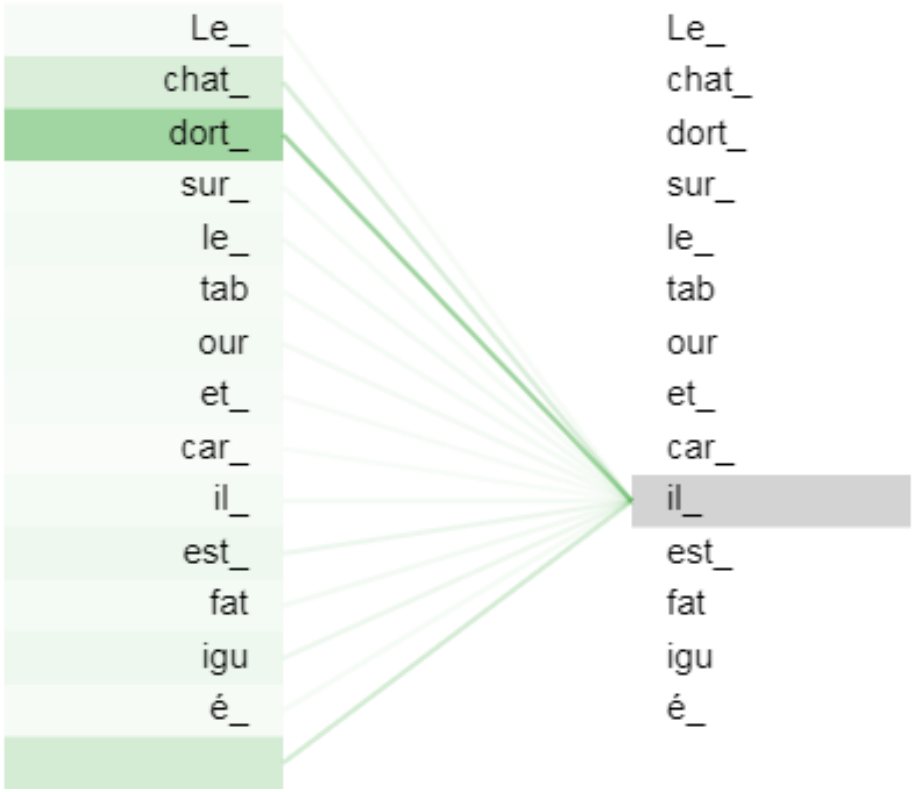
Attention weights for translation models.

Self-attention in transformer model

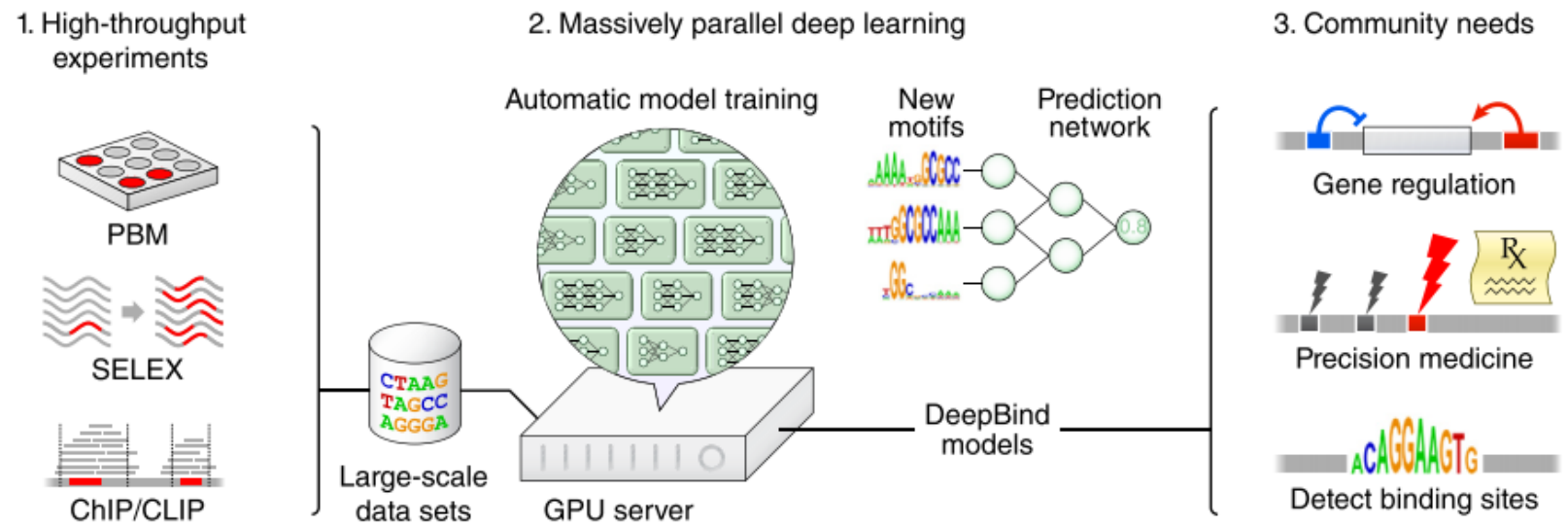
Self-attention is similar to attention, except that it is applied to the input sequence itself.

Self-attention allows to model long-range dependencies between any word in a sequence.

Self-attention is not directional as compared to RNN (LSTM or GRU), allowing parallel computing.



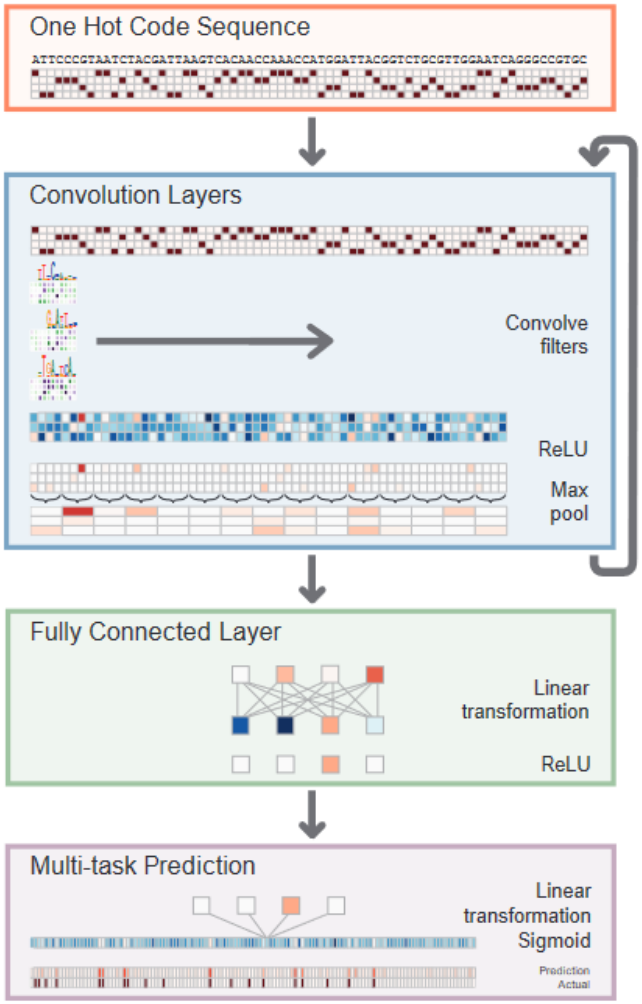
Example: Deepbind



- Predict binding proteins to DNA given the DNA sequence.

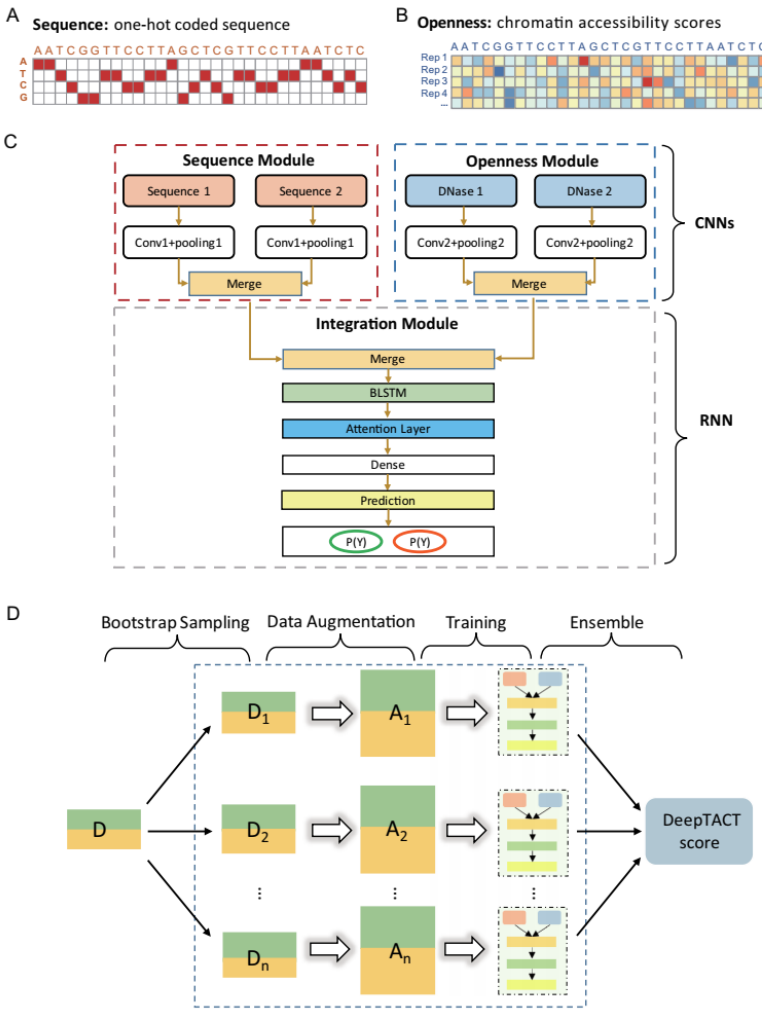
Example: Basset

- Predict chromatin accessibility (DNase-seq) from DNA sequences.

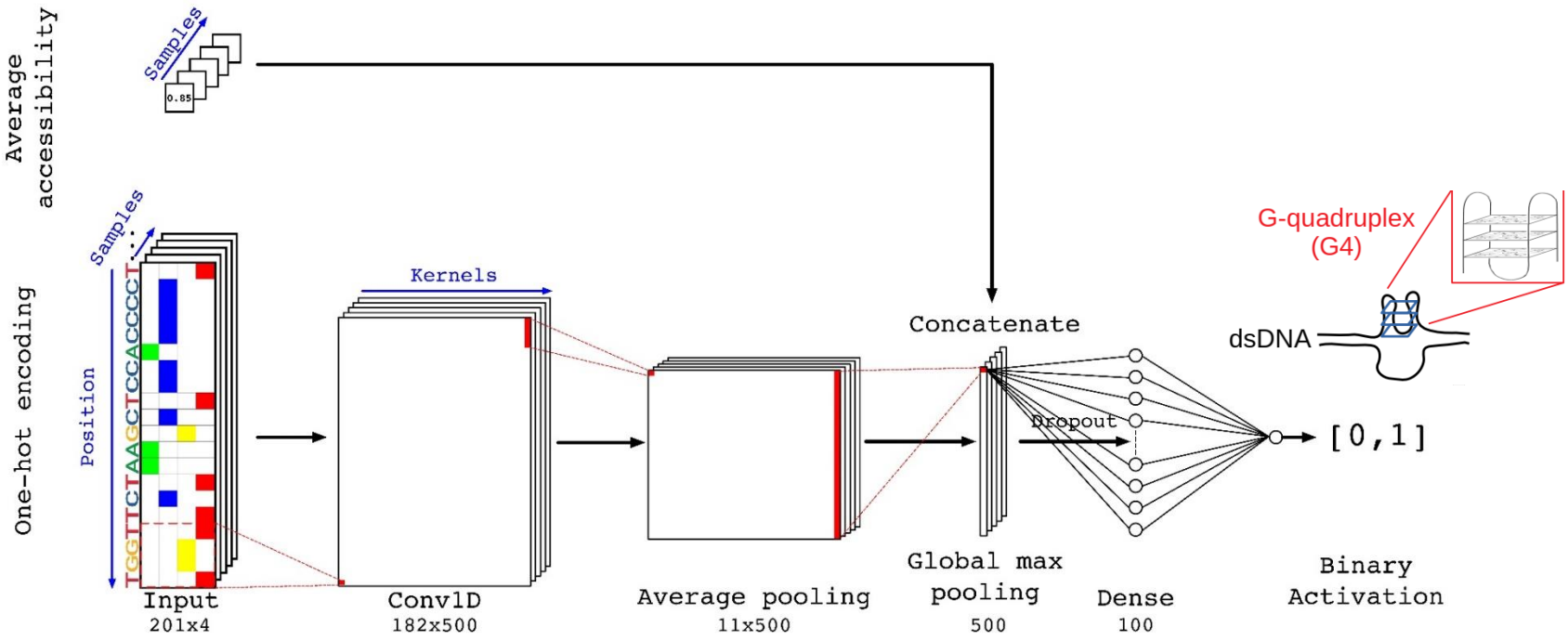


Example: DeepTact

- Predict long-range contacts (Hi-C) from DNA sequences and chromatin accessibility.



Example: DeepG4



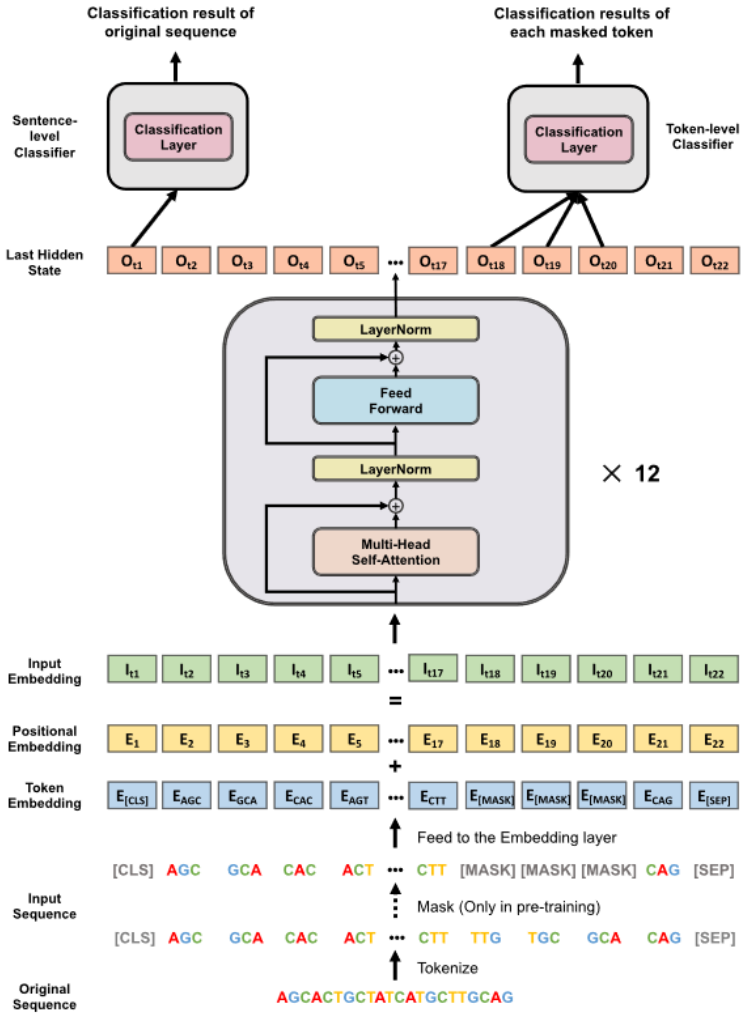
- Predict cell-type specific G-quadruplex structures given the DNA sequence and chromatin accessibility.

Rocher, Genais, Nassereddine and Mourad. PLOS Comp Bio 2021.

Example: DNABERT

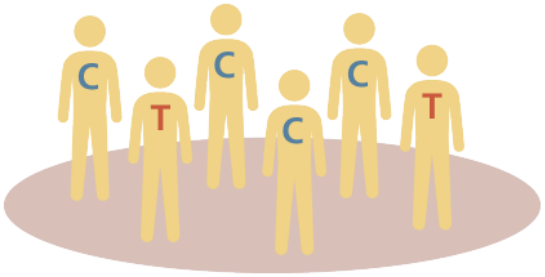
- The self-attention model DNABERT is trained by masking some kmers in the DNA sequence and then by trying to predict them using the other k-mers in the DNA sequence (context).

- At the end, the model provides features that encode DNA sequences in a very efficient way for any predictive task.

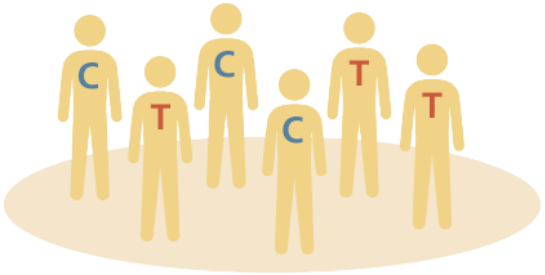


PREDICTING THE IMPACT OF SNPS USING DEEP LEARNING

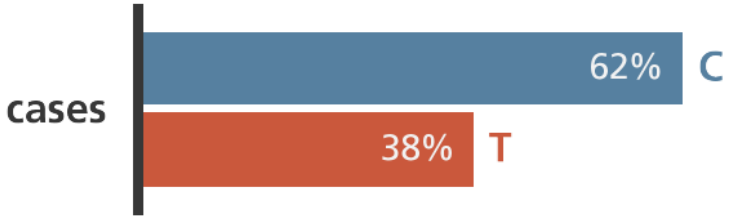
Genome-wide association studies and SNPs



cases (n=1,000)
people with heart disease



controls (n=1,000)
people without heart disease



SNPs affecting molecular phenotypes : gene expression, protein binding, …

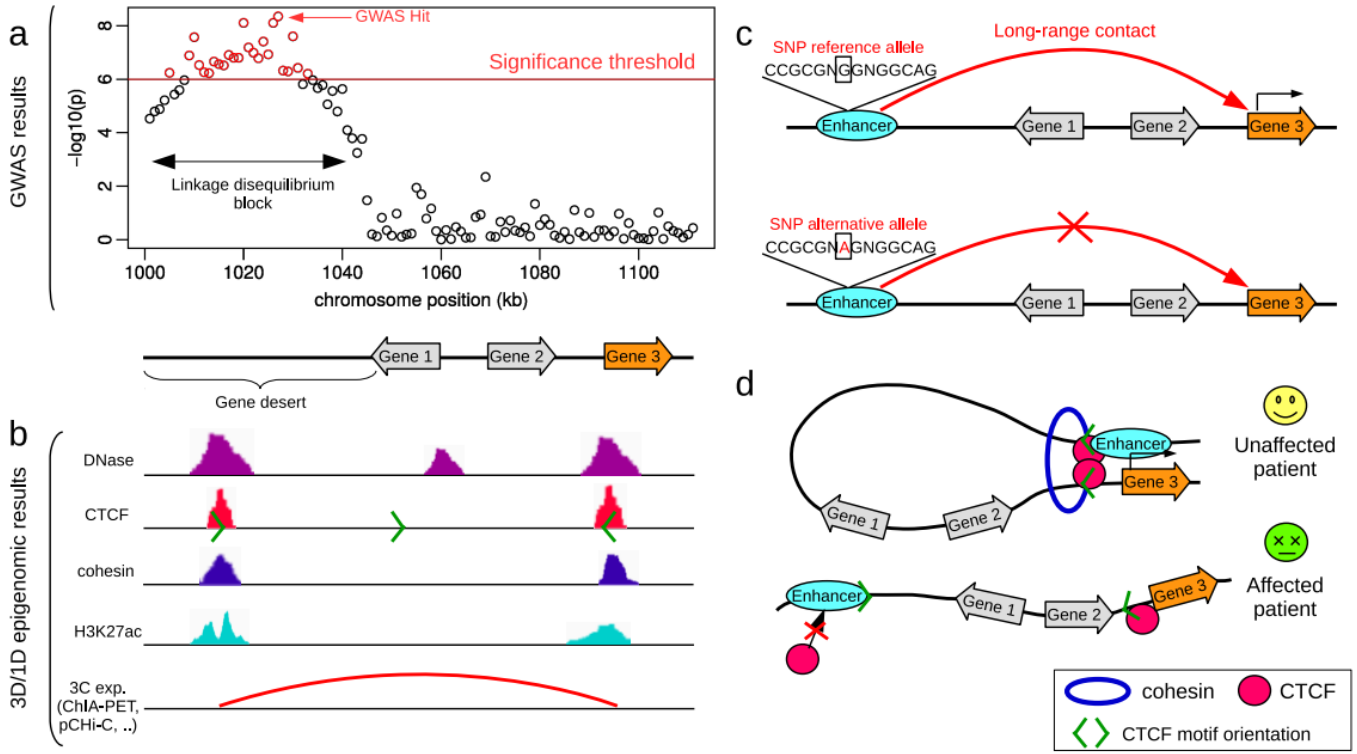


Figure 4. Post-GWAS analysis of risk loci using 3D genome information. a) GWAS mapping of single nucleotide polymorphisms (SNPs) associated with a complex disease. b) Mapping of epigenomic features including DNase, CTCF, cohesin, H3K27ac and 3C-based experiments. Green arrows represent CTCF motif orientations. c) 3C-based experiment results depending of the SNP allele. d) Functional model explaining the non-coding SNP effect.

Predict the impact of mutations